REMARKS

In summary, claims 1-21 are pending, having previously canceled claims 33, 47-48, and previously withdrawn claims 22-32, 34-46, and 49-73 under a restriction requirement. Claims 1 and 21 are independent. Claims 1, 2, 8, 10, 12, and 21 are rejected under 35 U.S.C. § 102. Claims 3-7, 9, 11, and 13-20 are rejected under 35 U.S.C. § 103. Claim 1 is herein amended without adding new matter. Applicants respectfully traverse the rejections. Reconsideration in view of the foregoing amendment and following remarks is respectfully requested.

Telephone Conversation With Examiner

Examiner Brown is thanked for the telephone conversation conducted on December 12, 2008. Proposed amendments were discussed. Cited art was discussed. It appears that the proposed amendments overcome the rejections based on the cited art.

Claim Amendments

Although unnecessary to overcome the cited art, claim 1 is once again amended to move prosecution forward to allowance as quickly as possible. In support of the amendment, see, e.g., pages 11 and 12 of the application.

> 1. (Currently Amended) A method of generating a computer program storable on a computer-readable medium, said method comprising the acts of:
>
> identifying a set of actions that are performed in the course of using a cryptographic algorithm to apply a cryptographic key to first data;
>
> identifying attributes of said cryptographic key corresponding to said set of actions;
>
> generating a first set of computer-executable instructions which includes instructions to perform actions functionally equivalent to said set of actions using said attributes but not said cryptographic key; and
>
> including said first set of computer-executable instructions in said computer program, wherein said computer program is operable to perform said ~~set of~~ functionally equivalent actions ~~independent of a whole and any part of said cryptographic key and without exposing or accessing any part of the~~

~~cryptographic key~~ without access to, storing in memory, or exposing a whole or segment of said cryptographic key.

Independent Claim 1, as amended.

Claim 1, as previously or presently presented, does not read on the cited art in any combination. None of the references identify attributes of a key corresponding to a set of actions performed when the key and algorithm are accessible. None of the references generate functionally equivalent actions based on the attributes but not the key. None of the references perform the functionally equivalent actions without access to, storing in memory or otherwise exposing a whole or segment of the key.

The Office Action made the same rejection against previously presented claim 1. The Office Action argues that performing actions (otherwise performed by an algorithm applying a key to data) without access to the key is equivalent to performing the actions with access to the key. Specifically, the Office Action argues that a property or attribute of a key is equivalent to an actual segment of a key (such as where the Aucsmith reference divides a key into segments and uses the segments instead of the whole key). This is essentially a statement of disbelief of the described and claimed subject matter. As explained in the application and previous remarks reiterated herein, Applicants respectfully submit that the argument in the Office Action is unsupportable. In contrast to claim 1, the main reference clearly accesses and uses segments of a key with a segmented algorithm. In any event, Applicants have once again amended the claims to make what they see as a misinterpretation moot.

Since the Office Action repeats the previous Office Action, Applicants repeat the previous Reply below, although the rejection is believed to be rendered moot. Accordingly, Applicants respectfully request prompt withdrawal of the rejection.

Reply to Examiner's Response to Previous Remarks

The previous Office Action comments that, "[i]t is unclear to the examiner how the computer program is able to perform the actions independent of 'any part of said cryptographic key' when it is employing 'key attributes' taken from said cryptographic key. . . . The examiner asserts that the applicant is merely separating the functions of the key into several actions taken by the code. . . . The examiner encourages the applicant to explain how taking key attributes and coding them into functions is different from breaking a key down into elements and coding them into functions." (Office Action, pp. 2-3).

It is respectfully submitted that this comment mischaracterizes the claimed subject matter and the prior art due to a perception that they are similar despite the actual disclosures of the claimed subject matter versus the prior art. The Office Action appears to confirm this by indicating that the Examiner does not believe or understand the claimed subject matter or Aucsmith. Office Action, p. 2 ('Aucsmith may state that it is using secret 'subparts'. However, Aucsmith [i.e. U.S. Patent No. 5,892,899, issued to Aucsmith *et al.*] appears to be doing what the applicant is also doing. It is unclear how the [claimed] program is able to perform the actions independent of 'any part of said . . . key'").

Aucsmith concerns partitioning a program and secret and reassembling them by repetition. Aucsmith states that "[t]he secret . . . is 'partitioned' into subparts 101, and program 100 is unrolled into a number of subprograms 102 that operate with subparts 101." Aucsmith, col. 3, ll. 52-55. The Office Action cites an example provided by Aucsmith at column 3, lines 60-67. Therein, Aucsmith states that the program is A = X * 8. The secret (8) is partitioned into four parts of 2 and the program is partitioned (unrolled) to run four times to operate on the four parts of the secret, i.e., A = A + (X *2). The result of running the subprogram four times on the four parts of the secret (i.e. rolling-up the subprograms into the program), repeatedly adding A to itself , is A = X * 8, which is the original program and secret reassembled. Aucsmith is dealing with the parts and the whole of the program and secret just as it says it is. As a result, the secret

and program are reassembled and exposed by running the last subprogram to roll-up the program.

In contrast, the claimed subject matter comprises a "computer program . . . operable to perform said functionally equivalent actions without access to, storing in memory, or exposing a whole or segment of said cryptographic key." Application, Independent Claims 1 and 21. The claimed subject matter doesn't store the key or any part of it. Instead, it may store only properties about the key or part of the key or, alternatively, use dynamically created code to represent the properties. Application, p. 16, ll. 24-30 (the inventive program can store properties of the key or "[a]lternatively, instead of storing any information about the [key], the program could . . . represent it in memory by dynamically creating code that . . . produces [the property], where any portion of the program that needs to know the [property] simply executes the dynamically-created code instead of retrieving the [property] from memory. This is a simple example of how a program can be constructed to use a [key or part of a key] without storing the [key or part of a key] in memory or otherwise exposing the [key or part of a key] to discovery by a user.").

In response to the Examiner's request for explanation about the claimed subject matter, Applicants refer the Examiner primarily to page 16, line 9 through page 18, line 13 of the Application, quoted below:

> An understanding of how cryptographic code generator 412 can generate code that applies keys 248 where the code does not have access to keys 248 begins with the mathematical principle that some computations involving a given number can be performed without directly using the number, but merely by using certain properties of the number. For example, one can determine whether a decimal integer is even or odd without knowing its entire value, but merely by knowing its least significant digit: a decimal integer is even if and only if its least significant digit is 0, 2, 4, 6, or 8 (or, in the case of a binary integer, if and only if its least significant bit is 0). One can determine whether a number is negative or non-negative without examining the entire number, but merely by examining the sign of the number. The number's least significant digit (or bit) and its sign are "properties" of a number, which, in the foregoing examples, may be all that one needs to know about the number in order to compute the information desired.

Thus, a program might receive a secret number as input, where the program performs a first action if the number is negative and a second action if the number is non-negative. In this example, the program could be constructed to store only the sign bit of the number. Alternatively, instead of storing any information about the number, the program could read the sign bit and then represent it in memory by dynamically creating code that (non-obviously) produces either 0 or 1 depending on what the sign bit is, where any portion of the program that needs to know the sign bit simply executes the dynamically-created code instead of retrieving the sign bit from memory. This is a simple example of how a program can be constructed to use a number without storing the number in memory or otherwise exposing the number to discovery by a user.

Analogously, cryptographic code generator 412 of the present invention makes use of mathematical properties of a particular cryptographic key 248, and creates code that computes the decrypted message that results from applying key 248 to a given ciphertext message without actually using key 248 itself or otherwise requiring access to key 248. Cryptographic code generator 412 similarly creates code that uses key 248 to validate cryptographic signatures – again, without requiring access to key 248. . . .

FIG. 5 shows the detail of an exemplary cryptographic code generator 412. Cryptographic code generator includes a key analysis module 504 and a code producing module 508 that produces the actual code that applies key 248. The key 248 obtained from key generator / key database 448 is received by cryptographic code generator 412 for analysis by key analysis module 504. Key analysis module 504 performs an analysis of key 248 in light of both its numerical properties and the cryptographic algorithm that will be used to apply it to a ciphertext message (or, in the case of authentication, to apply it to a digital signature). Key analysis module may identifies one or more properties or "attributes" of key 248 and produces or identifies one or more actions and/or functions 512 that would be performed by a cryptographic algorithm in the course of applying a key 248 having the identified attributes. For example, suppose that key analysis module 504 determines that the forty-second bit in the binary representation of key 248 is a one (i.e., "on"), and key analysis module 504 is programmed with information that a particular action/function 512 is always performed by a particular cryptographic algorithm whenever that algorithm applies a key whose forty-second bit is a one. This action/function 512 can be identified by key analysis module 504 and provided as input to code producing module 508. Key analysis module 508 may identify any number of attributes of key 248, and may provide all of the actions/functions 512 corresponding to these attributes to code producing module 508.

Code producing module 508 receives the actions/functions 512 that it will need to perform in order to apply key 248. It will be appreciated by those skilled in the art that it is possible to write a large – possibly infinite – variety of

> code that performs a given action or function 512. It will moreover be appreciated by those skilled in the art that, given a particular action or function 512, it is possible to generate different code to perform that function where the particular code generated is based on factors other than the action or function itself. In the example shown in FIG. 5, these "other factors" include hardware ID 224 and/or random number 432. Specifically, for each action or function 512, code producing module 508 produces code that performs such action or function 512, where the particular code produced is based on hardware ID 224 and/or random number 432.

Application, p. 16, l. 9 – p. 18, l. 13.

Determining the properties/attributes of a key, such as whether a particular number is odd or even, positive or negative and determining the functions performed by an algorithm that applies the key to data in order to create a program independent of both the key and the algorithm is patentably distinct from keeping the key and algorithm, but partitioning them so the algorithm operates repetitively on parts of the secret to reassemble the secret and algorithm per Aucsmith.

The claimed subject matter clearly does not read on Aucsmith. In the claimed subject matter, the program has nothing to do with the key or any part of it. It doesn't have access to it and doesn't expose it in whole or in part. As recited in the specification, attributes of the key and the functionality of the algorithm using the key are observed and utilized so that no part of the key is exposed in memory. There is no simple partitioning and reassembling of the key and algorithm as in Aucsmith, which exposes the key and its parts in memory.

Even though Applicants believe the claimed subject matter as previously presented in patentable over the prior art, Applicants have proposed an additional amendment otherwise expressing the distinction.

Rejection of Claims 1, 2, 8, 10, 12 and 21 Under 35 U.S.C. § 102

Claims 1, 2, 8, 10, 12 and 21 are rejected under 35 U.S.C. § 102(b) as being anticipated by U.S. Patent No. 5,892,899, issued to Aucsmith *et al.* (hereinafter referred to as "Aucsmith"). (Office Action, pp. 3-4.) Applicants respectfully traverse the rejection.

Aucsmith does not teach or suggest at least a "computer program . . . operable to perform said functionally equivalent actions without access to, storing in memory, or exposing a whole or segment of said cryptographic key." Claims 1 and 21. Likewise, Aucsmith does not teach or suggest the proposed amendments to independent claims 1 and 21.

All of the foregoing remarks apply equally well to the present rejection of claims 1, 2, 8, 10, 12, and 21. As previously pointed out, Aucsmith clearly operates on (i.e. is dependent upon) the parts and the whole of a secret and algorithm, disassembling and reassembling them, thereby exposing the secret in whole and in part.

Claims 1, 4, 7, and 10 of Aucsmith confirm Aucsmith's dependence on the key, in whole and in part. In its Claim 1, Aucsmith states that "programming instruction blocks operat[e] on corresponding subparts of a secret distributed among them," while in its Claim 4 Aucsmith states that a "method for executing a program that operates on a secret . . . compris[es] . . . executing a first unrolled subprogram of the program . . . operating on a first subpart of the secret; and executing a second unrolled subprogram of the program . . . operating on a second subpart of the secret." Aucsmith, Claim 4.

The presence of the dispersed key and reassembly of it by running the last subprogram remains problematic for security. In contrast, the claimed subject matter creates a program independent of the key, in whole and in part, that operates without exposing the key, in whole or in part, providing greater security than Aucsmith.

Thus, it is believed that independent claims 1 and 21 are allowable over Aucsmith. At least in view of their dependence on claim 1, it is similarly believed that claims 2-20 are also allowable

over Aucsmith. Applicants respectfully request reconsideration of claims 1-21, as amended, and withdrawal of the rejection of claims 1, 2, 8, 10, 12 and 21 as being anticipated by Aucsmith.

Rejection of Claims 3-7, 9, 11 and 13-20 Under 35 U.S.C. § 103

Claims 3-7, 9, 11 and 13-20 are variously rejected under 35 U.S.C. § 103(a) as being unpatentable over Aucsmith in view of one or more of the following additional references: U.S. Patent No. 6,643,775, issued to Granger *et al.* ("Granger"); U.S. Patent No. 6,715,079 issued to Maytal ("Maytal"); U.S. Patent Application Publication No. 2002/0178412 in the name of Matsui ("Matsui"); U.S. Patent No. 5,912,972, issued to Barton ("Barton"); U.S. Patent No. 6,138,236, issued to Mirov ("Mirov") and U.S. Patent No. 5,758,293, issued to Frasier ("Frasier"). (Office Action, pp. 5-10.) Applicants traverse all rejections.

All foregoing remarks apply equally well to the rejections of dependent claims 3-7, 9, 11 and 13-20 under 35 U.S.C. 103. Accordingly, it is requested that the rejection of claims 3-7, 9, 11 and 13-20 under 35 U.S.C. 103 be reconsidered and withdrawn.

Amendments made herein as well as those previously made are without abandonment of subject matter. Applicant expressly reserves the right to, in the pending application or any application related thereto, reintroduce any subject matter removed from the scope of claims and introduce any subject matter not present in current or previous claims.

CONCLUSION

In view of the foregoing amendment and remarks, it is respectfully submitted that this application is in condition for allowance. Reconsideration of this application and an early Notice of Allowance are requested. In view of the pendency of this application for more than seven years, the Examiner is cordially invited to contact the undersigned representative of Applicants for any reason that Examiner believes may speed up allowance of the application.

Date: December 22, 2008

/**Joseph F. Oriti**/
Joseph F. Oriti
Registration No. 47,835

Woodcock Washburn LLP
Cira Centre
2929 Arch Street, 12th Floor
Philadelphia, PA 19104-2891
Telephone: (215) 568-3100
Facsimile: (215) 568-3439